第2章

テスト・エンジニアの悩み相談室

── テスト計画とテスト管理に関する10の疑問

大西建児

ここでは，プリンタ，コピー，ファクシミリの複合機の開発プロジェクトをイメージしながら，テスト計画やテスト管理の課題について，Q&A形式で解説する．主人公は，今回初めてテスト・チームが結成されたOA機器メーカの新米テスト・チーム・リーダである．彼は，これまでウォータ・フォール型プロセスに基づくソフトウェア開発しか経験したことがなく，今回初めて反復型の統一プロセスを経験することになったのだが….　（編集部）

ハードウェアでは，モノを製作する途中段階から完成までを通じて，製品の品質に問題がないかどうかを目視や動作確認によって検査することはあたりまえでしょう．組み込みシステムのソフトウェアを開発する場合も，品質を確認する手続きを踏むことはあたりまえなのですが，ハードウェアを検査するように，組織的かつ体系的にテストに取り組んでいる組み込みシステム開発の現場は少ないように見受けられます．

組み込み機器をハードウェア（機構系＋回路）主体で製作していた組織で，時代とともに機器の制御部をハードウェアからソフトウェアへ置き換えていったところでは，どうやらソフトウェアの特性を理解したうえでテストに取り組めていないケースが多いと筆者は感じています．組み込み機器をハードウェアとソフトウェアで構成される「組み込みシステム」ととらえるなら，製品を検証する際に，ハードウェアとソフトウェアに対して，“個別”および“組み合わせ”の両方の観点から対処することが必要となります．

本稿では，以下のようなエンジニアを対象に解説します．

- ハードウェアの検証は得意だが，ソフトウェアについてはどのようにテストに取り組むのか暗中模索している方
- これからソフトウェアを含めた品質保証に取り組む方

また，組み込みシステムにおいてソフトウェアのテストをプロセスとしてとらえ，マネージメント（管理）していく手法について説明します．とくにテスト計画とテスト管理のプロセスを中心に，仮想事例を使って紹介していきま

株式会社ComponentQuality（CQ社）
設立20年：本社 東京都豊島区巣鴨 某所
従業員数：国内15,000人，海外10,000人規模
概要：国際的にも認知されだした．最近では中国にも進出し，オフショア開発（システム開発の海外企業への委託）にも取り組む．設立時はラベル・プリント機の開発から始まったが，ワープロからプリンタやスキャナの開発へ発展し，現在の主力製品は複合機（MFP：multi function peripherals）となっている．

● **組織体制**

大きくは，全社スタッフ部門，一般消費者向け部門，企業向け部門，デバイス特化部門の四つ．

1）スタッフ部門
- 企画部
- 購買部
- R&D統括本部
  - 基盤テクノロジ開発ラボ
  - プロダクト・マネージメント推進室
  - 品質保証統括室

2）コンシューマ・エンターテインメント・カンパニ（CEC）
- ラベル・プリンタ
- 家庭向けプリンタ
- 家庭向けスキャナ
- 家庭向け電話機

3）オフィス・ネットワーク・カンパニ（ONC）
- オフィス向けディジタル複合機
- オフィス向けIP電話機
- 電子出版システム

図1　仮想事例のプロット（背景）
今回の仮想事例は筆者による完全なフィクションである．

す[注1].

今回の仮想事例のプロット（背景）を**図1**に示します.

1 仮想事例：テスト計画編

季節は2005年春．開発コード「かもねぎ」プロジェクトが立ち上がってから約1年が経過しました．2004年の春にプロジェクトがキックオフしてから，企画部による企画検討，製品戦略の策定などに始まり，R&D統括本部を中心に製品マネージメントの基本方針および製品開発全体スケジュール（いわゆるロードマップ）の確定と，計画作業は着々と進んできました．

製品開発全体スケジュールの策定段階からは，プロダクト・マネージメント推進室が主管となり，「かもねぎプロジェクト」チームが各事業部をまたがる組織体として結成されました．このような事業部門をまたがった大規模開発プロジェクトは，CQ社としては初の試みでしたが，経営トップからの「ハードウェア/ソフトウェアともに，CEC/ONCで共通化したプラットホームの構築に取り組む」という方針に基づき，プロジェクトは動き出したのです．

2004年の秋ごろからは，実際に製品を開発するCECとONCのプロジェクト・マネージメント担当も加わり，ファースト・リリースに向けての具体的なプロジェクト計画の策定が始まりました．プロジェクト計画の策定では，ソフトウェア開発に適用する開発プロセス・モデルの検討が行われました．従来の開発では，CECもONCも開発プロセス・モデルはウォータ・フォール型で，こつこつと，しかし，かなりばたばたとした開発を行っていました．世の中にはウォータ・フォール型より進化した開発プロセス・モデルがあることを関係者は知っていましたが，これまでほ

4）デバイス・デベロップメント・カンパニ（DDC）
- 産業用プリンタ機器
- 周辺デバイス機器

● 対象：ディジタル複合機の新製品開発プロジェクト

開発コード「かもねぎ」

プロジェクトの全体ゴール
- 最新のトレンドを反映し，かつ独自の特徴を持つ，革新的なディジタル複合機を開発する
  キャッチ・コピー：「もっとスゴくてイチバンきれい」
- 魅力的な新機能を早く市場へリリース
  - セキュリティ機能強化
  - ネットワーク連携機能
- 同一価格帯の複合機で，読み取り画像，プリントアウト品質ともにユーザ層による官能評価でいちばんを目指す
- ハードウェア，ソフトウェアともに，CECとONCで共通化したプラットホームの構築に取り組む

スケジュール

0）プロジェクト・キックオフ：2004年4月
1）企画部によるプロダクト全体構想策定：2004年4月～8月
2）企画部と基盤テクノロジ・ラボによる製品戦略検討および策定：2004年9月～12月
3）プロダクト・マネージメント推進室主導で，CEC/ONC開発部門マネージャ・クラスを含めて全体プロジェクト計画検討および策定：2004年11月～2005年3月
4）製品開発：2005年4月～2006年7月
※製品の市場投入時期：2006年7月

● 登場人物



秋田浩大（あきた・こうだい）<主人公>
［ONC所属 テスト・チーム・リーダ］
明朗快活，ノミュニケーションは得意.



山中和俊（やまなか・かずとし）
［プロダクト・マネージメント推進室所属 特命チーム・マネージャ］
つい最近までは秋田と同じONCにて，数々の伝説的な製品を世にはなった敏腕マネージャ．プロジェクト/製品マネージメントに造けいが深く，経験も豊富なため，プロダクト・マネージメント推進室設立メンバとして参画し，現在に至る.



大山晋平（おおやま・しんぺい）
［品質保証統括室 室長］
元すご腕だったというウワサだが，現在は現場に来ていろいろと品質に関するアドバイスをしている．CQ社のQMS（品質管理システム）構築を取りまとめる立場にある．ソフトウェア工学にも，品質分野にも造けいが深いのだが，たまにわからないこと（本人いわく，高尚らしい）を言うため，だれも議論についていけなくなることがあるらしい．なぜか憎めないキャラクタである．ノミュニケーションが大好き.

注1：本稿を作成するに当たっては，アイデア出しの段階から内容のレビューに渡り，ソフトウェアテストシンポジウム実行委員会などにてお世話になっている富士ゼロックスの秋山浩一さんと，飲み仲間でもあるD.I.Y（匿名）さんに全面的にご協力いただいた．ほんとうにありがとうございました.

かのモデルを試すスキルも余裕もありませんでした．

今回のプロジェクトでは，新しい試みを行うには絶好の機会であるため，いろいろと検討した結果，統一プロセス（UP：Unified Process）を開発モデルに適用してプロジェクト計画を策定することで，合意に至りました．

このモデルを適用したおもな理由は次のとおりです．

- 大規模な開発となるため，計画駆動型の開発モデルが必要
- 機構系，回路，ソフトウェア全般で新規開発の要素が多く，開発を進めながら詳細を確定していける開発モデルがふさわしい
- プロトタイプを用いてアーキテクチャを十分に検証する必要がある
- 競合他社の動向をにらみながら，新機能の取捨選択をしつつ開発を進められるモデルにしたい

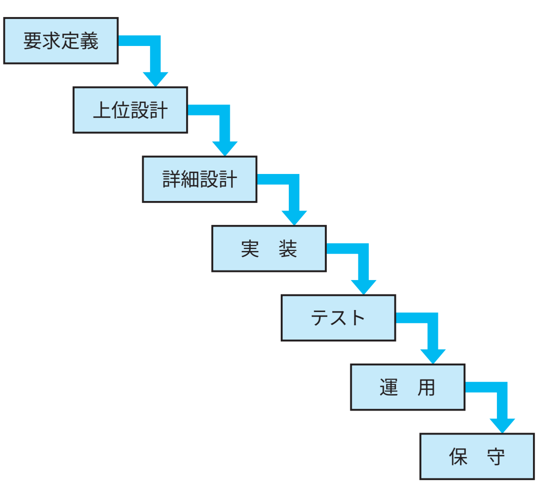
ここで，ウォータ・フォールと統一プロセスについて簡単に説明しましょう．

ソフトウェア開発のライフ・サイクルをモデル化したもので，もっとも古典的で一般的なモデルが，ウォータ・フォール型です．このモデルを適用した開発は図2（a）のように流れていきます．「要求定義」，「上位設計」など，それぞれの過程は工程やフェーズ，あるいはプロセスと呼びます．ウォータ・フォール型は逐次的なモデルであり，原則としてプロセスの後戻りは許されません．このため，開発の初期で詳細な開発プロジェクトの全体計画を立てる必要があります．

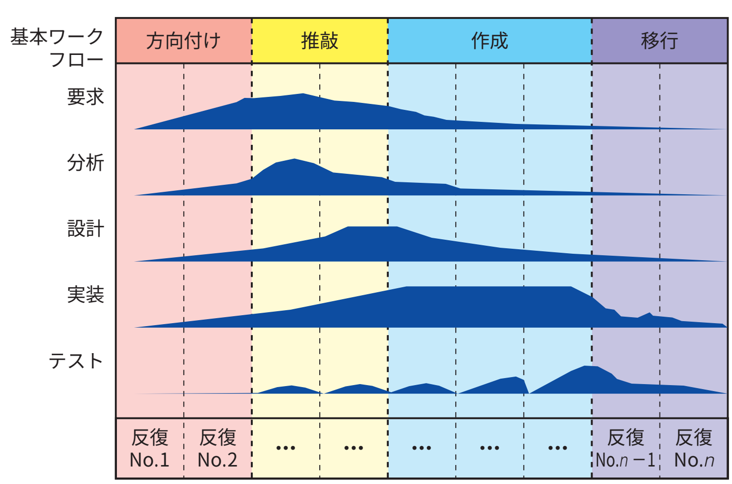
一方，統一プロセスでは，ソフトウェア開発プロセスを「ユーザからの要求をソフトウェア・システムに変換するために必要な一連の作業」としています．また単なるプロセスではなく，さまざまなソフトウェア・システムに適用できる汎用的なプロセス・フレームワークとして定義されています．開発サイクルとしては，図2（b）のように大きく「方向付け」，「推敲（すいこう）」，「作成」，「移行」の四つのフェーズで構成されます．

- **方向付け**――製品に対するニーズやシーズをもとに製品の構想をまとめ，企画を作ります
- **推敲**――製品のユースケースの大半を詳細定義して，システム・アーキテクチャを設計し，確定します．
- **作成**――推敲で定めたアーキテクチャに基づいて実製品を実装し，構築していきます．
- **移行**――製品がβ版のリリースに移行した時点から始まり，リリースからエンド・ユーザへの展開と進んでいきます．

各フェーズはさらに反復と呼ばれる計画の最小単位に分割され，それぞれ要求，分析，設計，実装，テストという流れで作業を進めます．このため，反復は「ミニプロジェクト」とも呼ばれます．統一プロセスでは上記のようなプロセスを適用することで，インクリメンタルな



（a）ウォータ・フォール



（b）統一プロセス

図2　開発プロセス

（a）のウォータ・フォールは，システムの開発手順を示す古典的なモデル．水が滝を流れ落ちるように開発を順に進めることからこの名まえが付いた．水が逆流しないのと同じく，基本的に直線的にプロセスが進む．後戻りが起きると，かなりの工数がむだになってしまうという特徴を持つ．（b）の統一プロセスは，プロセスにとどまらず，さまざまなソフトウェア・システムに適用できる汎用的なプロセス・フレームワークとして定義されている反復型の開発モデル．方向付け，推敲，作成，移行の四つのサイクルを持ち，各サイクルを反復と呼ばれる計画の最小単位に分割し，要求，分析，設計，実装，テストという流れで開発を行う．

（漸進的な）開発を可能としています．

2005年春となり，プロジェクト計画も大枠でまとまり，いよいよ本格的な開発フェーズである方向付けフェーズに突入しました．プロジェクト計画で定義した，「かもねぎ」のファースト・リリースに向けた開発スケジュールは，以下のとおりです．

- **方向付けフェーズ**：2005年4月～6月（3ヵ月間）
- **推敲フェーズ**：2005年7月～10月（4ヵ月間）
- **作成フェーズ**：2005年10月～2006年5月（7ヵ月間）
- **移行フェーズ**：2006年6月～7月（1ヵ月間）

今回，「プロジェクトかもねぎ」チーム全体のプロジェクト・マネージャ（PM）を務めることになった自他ともに認める敏腕PMの山中和俊は，これまで経験した数々の開発の中で，ソフトウェア・テストをないがしろにしたため，さまざまな痛い目にあったことを思い出しながら，なんとかできないものかと思案していました．

そこで，プロダクト・マネージメント推進室とほぼ時を同じくして設立された品質保証統括室の大山晋平にプロジェクト計画策定段階から相談していたのでした．

時計の針を巻き戻すこと少々，2004年の冬，とある居酒屋での会話です．鴨ならぬ，普通の「ねぎま」を山中と大山の二人がほおばりながら….

「大山さん．例の"かもねぎ"の全体計画を今練っているんだけど，品質を何とか開発の早期から確保できるようにしたいんです．大山さんとは過去のプロジェクトで何かとやりあった仲なんで，ここは一つ共同戦線を張りませんか」

「それはいいけれど，開発部隊でだれかテスト・フェーズのめんどうを見れるの？ これまで開発事業部内にテスト・チームの一つもないおかげで，最終的な製品のテストはすべてうちでめんどうみていたからねぇ．うちもサポートならできるけど，直接開発に入り込めるほど人も予算もないよ．たいへんなんだからもぅ．ぶつぶつ…」

「そう．そもそもテスト・チームがないと，今回のような開発規模だと破たんすると危ぐしているんです．で，ぼくとしても生きが良くてやる気のあるのを何人か集めて，チームとしてテストしてもらおうと考えてるんです．大山さん，リーダができるくらいテストにまじめに取り組んでいる若いやつをCECで挙げるとしたら，だれがいますか」

「最近，秋田君がテストの勉強をしたいと言って話を聞きにきたり，机の上にテスト・マネージメントの本を何冊か置いてたけど，彼なんてどう？ 何より覇気も元気もあるし，少々のことではぶっ倒れなさそうなのがテスト向きだと思うけど」

「なるほど，秋田君ですか．確かに彼なら人当たりもいいし，うまく開発メンバとも立ち回れそうです．じゃ，テスト・チームのリーダは彼に決まりとして，大山さん，彼のサポートを陰に日なたにお願いしてもいいですか」

「それはいいけれど，そっち（PM）もちゃんとフォローしてよね．そういえば別のプロジェクトのときもぶつぶつ…」

こうして秋田君にテスト・チーム・リーダとして白羽の矢が立ったのでした．

では，時計の針を2005年の春まで戻しましょう．

「さあ，いよいよテスト計画だ！」と張り切ってはみたものの，プロジェクト計画にはなじみがあっても，テストの計画では何をどう策定するべきか今ひとつピンとこない秋田君です．いちおうテストの書籍を読みながら独学はしてきましたが，自信が持てるというほどではありません．

テストはプロジェクトと同じように，まず計画したうえでテスト項目を設計してテストを実施するというサイクルを，テストの管理を軸に回していくということは，何となく理解できました（図3）．とはいえ，実際にテスト・プロセスを回すとなると，それなりに知識と経験がないと不安です．今回，心強いのは山中さんから「テスト・チーム運営でわからないことがあったら，何であれ品質保証統括室の大山室長に相談していいよ」と言われていることでした．

テスト・プロセスの構築に向けて，まず本プロジェクト

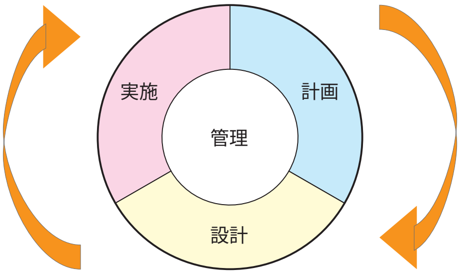


図3 テスト・サイクル

テストのライフ・サイクルを，四つのプロセスであるテスト管理，テスト計画，テスト設計，テスト実施で表現している．プロジェクト管理のPDCA（Plan，Do，Check，Act）で回すのと同じように，テスト・サイクルをテスト管理を軸に回していく．

のテスト計画を作成することになり，テスト計画書そのものはなじみのあるプロジェクト計画書をひな型とすることにしました．計画をまとめながら，わからなかったり不安に感じている箇所については，まとめて大山さんに聞くことにして，秋田君はテスト計画の策定を進めていきました．

以下では，テスト・チームのリーダである秋田君の疑問や不安に品質保証統括室の大山さんがコメントする形で，テスト計画の問題を説明していきます．

Q1．テストのWBSをどう定義するべきか？



「これまで開発部門内のテストを開発メンバ各自の責任で実施してきたため，実機を動かすシステム・テスト以降は品質保証統括室におんぶにだっこでした．まずは単体テストから結合テストまで，どんなことをやっているのか，何人かのメンバでWBS（work breakdown structure）[注2]を洗い出してみることにしました．ぼくから何人かのメンバにお願いして，それぞれのWBSをまとめてメールで連絡してほしいと依頼したんです．ところが，ほとんどのメンバから返事が来なくて，WBSをまとめられない状況なんです．どうしたものでしょうか」



「WBSを実際に書き出そうとして，案外うまく書き出せてないんじゃないかな．ふだん設計作業で行っている構造化を実際の作業に置き換えて表現すればいいだけなんだけど，なぜかWBSということばになると敷居が高いと感じてしまう人もいるようだ．そんなときは，メンバ全員を集めて，ブレーン・ストーミング（ブレスト）形式でタスクをすべて洗い出してから整理し直して，WBSと表現するという手段が使えるよ．

ポイントは，ブレストで現在行っている作業の構造は明らかにできるけど，そのままだとWBSとしては不十分だということを理解しておくこと．テスト・プロセスで行う作業全体を眺めたうえで，最適な作業の構造になるようにメンバどうしで検討してみてほしい．

注意点としては，ソフトウェア側の作業だけでなく，メカ（機構系）やエレキ（回路）とリンクする作業を明確にすることかな．作業の相関関係や作業の境界をわかるようにすることで，メカ，エレ，ソフトウェア，つまりシステム開発全体で製品を評価できるようなWBSをまとめられるからね．

そうそう，ISO 9000対応で作成した品質マニュアルで定義したプロセス・フローと矛盾が発生するのなら，品質マニュアル側のフォローも必要だからチェックを忘れないように．ただ，品質マニュアルが絶対なんじゃなくて，良いプロセスとすることが大事なんだから，無理に品質マニュアルに合わせようなんてしちゃいけないよ．CMMI（Capability Maturity Model Integration）で定義したプロセスがあると同じような問題が起こるけど，うち（CQ社）はまだCMMIには取り組んでないんで，とりあえず考慮しなくていい．まぁ，いったん作ったプロセスに縛られる必要はないよ」

Q2．テストで準備するべき環境は？



「テスト用にハードウェアのシミュレータを用意したり，開発用のビルドとテスト用ビルドは分けて構成管理できるようにするつもりですが，ほかに環境面で注意するべきポイントがあれば教えてください」



「意外とテスト直前になって用意できてなくて慌てるのが，周辺機材の準備だ．例えば，ハブやルータといったネットワーク周りの機材やネットワークのアドレス設定といったこと．イントラネットの環境だから，LANの同軸ケーブルが足りないなんてことも実際にあったな．ケーブルの数は足りてるんだけど長さが足りないとかね．ネットワーク周りだと最近の複合機ではセキュリティのテストも重要だから，そういったテストも可能な環境を用意しないとまずいだろうね．

実機テストの最中に特定サイズの用紙がなくなって，紙切れでテストが進まない，なんてこともあるから，消耗品は忘れずに十分に用意しておかないと．そうそう，消耗品（例えばトナー）は純正品以外にも検証が必要なら，サード・ベンダの製品も用意しておく必要があるよ．

それから，試作機の数が足りないことはいつも問題になるな．ハードウェア開発側とよく連絡を取って，試作機のしあがりタイミングとテストの実施がかみ合うようなスケジューリングが必要だよ．ハードウェアのシミュレーションも考えているようだから，実機とシミュレータの使い分けを効率良くできるような配慮が必要だね．

そうそう，笑い話じゃないけど試作機の数がそろったのはいいけど，直流電源が足りなくなって秋葉原に走ったこともあったっけ．くれぐれも環境の準備をおこたらないようにしないといけないよ」

テスト環境として用意する必要があるものを，**表1**に

注2：その名のとおり作業を構造的に示したもので，プロジェクト・マネージメント分野の用語．

表1　テスト環境として用意するもの[9]

分　類	概　要
テスト用スペース	開発者の机，テスト用機材を設置する棚．テスト担当者がテストを実施するオフィスや，テスト専用の部屋（テスト・ラボと呼ばれることもある）といったテスト実施用のスペース
テスト機材用電源	テスト対象システムに必要な交流電源や直流電源，停電対策のためのCVCF（電圧/周波数を安定化させた電源），UPS（無停電電源装置）といった電源設備
ハードウェア・テスト用設備	環境試験装置（温度や湿度を変動させる），電波暗室，耐圧試験装置，衝撃試験装置，耐久試験装置など，ハードウェアの信頼性にかかわるテストを実施するための設備
テスト実施支援設備	テスト実施の状況を測定したり，記録するための設備などを指す．音響測定機器やカメラやビデオ，遠隔テスト用のテレビ電話といった設備など

まとめます．

Q3．テストの工数を見積もるときの注意点は？

「テストの実施工数は，過去の製品開発でだいたい見えているので，それを参考に見積もるつもりですが，こういったやりかたをとるときに何か気をつけておくことはありますか？」

「基本路線は，過去のデータを参考にすることでいい．ただ，これには落とし穴があってね．だいたい見積もりには不ぐあい注3対応の時間が含まれていないんだ．バグがゼロの製品なんてありえないから，見積もり段階から不ぐあい対応の工数や手戻り工数も見込んでおかないといけない．

そうそう，実機結合テストやシステム・テストに突入する前にテスト・チームによる受け入れテスト工数を確保しておくといい．いきなりテストに突入して，ハードウェアの不ぐあいなんだか，ソフトウェアの不ぐあいなんだかわからなくて右往左往する，なんて話をよく聞く．そんなことがないように，対象となるテスト・フェーズで確認する基本機能に問題ないことを，受け入れテストで事前確認するようにしたらどうだろう．

それから，今回の製品にはリモート課金用の新機能が追加されているから，そのテストにはほかの不ぐあい対応工数よりも多めに，うーんそうだなぁ，これまでの傾向だと3倍くらい工数にバッファを置いておいたほうが無難だろうね」

Q4．テスト・チームのアサインをどうすべきか？

「今回，テスト作業の一部を中国の系列会社に出すことで，コストを削減するという方針がでています．そういうのを含めて人のアサインで今の段階から気をつけることって何ですか」

「中国もそうだけど，これまでパートナさん（協力会社や派遣会社から来る技術者をCQ社ではこう呼ぶ）にテストをお願いすることでいろいろと問題があったよ．例えば，実機評価をパートナさん中心で進めていた機種のテスト・ノウハウはその機種でしか通用しなかったり，ノウハウが人にくっついてしまって，パートナさんのある技術者がいなくなっちゃうとテストの効率がぐっと落ちてしまうこともあった．

そうならないためにも，テストのプロセスを標準化してまとめることには意味があるんだ．とくに，海外の会社にお願いする場合，テストの進捗を直接確認できないからこそ，プロセスで管理しておかないと，何をテストしているのか，実際にテストが進んでいるのかが日本から見えなくなってしまうよ．

これは山中さんとも相談して決めたんだけど，プロジェクト計画全体や開発プロセス，テスト・プロセスといったたいせつなことは直接中国に行って説明することにした．もちろん，秋田君にも同行してもらうよ．プロセスの説明以外にも，どうやってテストのノウハウを標準化して文書化したりトレーニングしていくかを検討しなくてはいけないと認識しているんだ．

テスト・フェーズを安易にパートナさんやオフショア先である中国に出すと，製品評価のノウハウが失われかねないから，この点はとても重要だと理解しておいてね」

Q5．UPでテスト計画をまとめるときのポイントは？

「これまでのCQ社の開発モデルは，ウォータ・フォール型でしたから，統一プロセスのように漸進的に回していく開発モデルだと，プロジェクト初期段階でどれだけテスト計画を詳細に立ててよいか判断が難しいんです．何か目安はないですか」

注3：不良や欠陥と同義と思われるかもしれない．しかし，不良や欠陥と表現してしまうと，作り手側に瑕疵（「かし」と読む．行為・物・権利などに本来あるべき要件や性質が欠けていること）責任が発生する．そうなれば，無償対応やリコールへとつながる．「不ぐあい」ということばは，あくまでも「ぐあいが悪い」ということであって，瑕疵があることを認めているわけではないという意味も持っている．



「統一プロセスでは，推敲フェーズが終わるまでプロダクトの全体像が見えてこないから，とくに作成フェーズ以降のテストの詳細を検討できないのはあたりまえだと思う．だから統一プロセスでは，開発全体を通じたテスト戦略や方針，大まかなスケジュールなどは，マスタ・テスト計画という形で表現するべきだろう．マスタ・テスト計画で，大枠レベルでテストに必要な人的リソースやスケジュール，コスト，環境などを明らかにしておいたうえで，各フェーズのテスト計画を作成するようにすると整理がしやすくなるからなんだ．

これを文字どおりに受け取れば，マスタ・テスト計画書の下位文書として，方向付けフェーズ・テスト計画書，推敲フェーズ・テスト計画書，作成フェーズ・テスト計画書，移行フェーズ・テスト計画書の四つが必要になっちゃうよね．ただ，方向付け段階は実機を使ったテストというより，論理的なテスト（レビュー）が主となるから，一つの独立したテスト計画書を作成する手間をかける必要はおそらくないだろう．現実的には，方向付けフェーズのテスト計画は，マスタ・テスト計画の記述レベルで今回の開発では十分だろう．

プロジェクトにせよテストにせよ，計画は1度作ったら，それでおしまいではないということを肝に銘じておくこと．今回のようにプロジェクトの早い段階から計画をまとめるのが大事なのはもちろんだけど，開発の進捗に合わせて計画を適宜見直さないと，“生きた計画”として使えないことを決して忘れないようにね」

ほかにも秋田君は大山さんにいろいろと相談しながらも，何とかマスタ・テスト計画の草案をまとめ上げたのでした．**リスト1**に，テストにかかわる文書の標準であるIEEE std 829-1998 Standard for Software Test Documentに定義されているテスト計画書の項目を示します．ウォータ・フォールであれ，統一プロセスであれ，テスト計画書の基本構造は変わらないので，参考にしてください．

2 仮想事例：テスト管理編

マスタ・テスト計画の草案をまとめてから，数ヵ月がたった2005年秋．

春から夏にかけて方向付けフェーズが実施され，ついに「かもねぎプロジェクト」は推敲フェーズへと移行しました．システム・アーキテクチャが徐々に明らかになってきています．いよいよ本格的なテスト項目の作成，そしてテストを実施する段階になりました．

秋田君は，大山さんのアドバイスに従って，フェーズごとのテスト計画については推敲フェーズ以降のものを策定しました．この詳細テスト計画に基づいて，テスト設計（テスト実施項目やテスト方法の検討，それに基づくテスト項目の作成）を行いました．推敲フェーズも2回目の反復に入り，「かもねぎ」のアーキテクチャを検証するために，シミュレーション・モデルを使ったテストが始まりました．

本格的なテストが始まってからというもの，秋田君はテスト・チームをマネージメントしてるんだという実感が日々強まってきます．そのような中，テスト・チームのマネージメントについて，秋田君は大山さんにアドバイスを求めていました．

Q6．どうなっていればテストは順調？



「今はとりあえずテストの実施項目数を見て，計画した予定消化項目数と比較することで進捗を把握するようにしていますが，どうもこれだけだと不安なんです．というのも，同じ数のテスト項目を消化したとしても不ぐあいの出方しだいで，順調かそうでないか，何となく

リスト1
テスト計画書の項目
テスト計画書の記載内容の詳細を知りたい方は，稿末の参考文献をはじめ，テスト管理の説明のある書籍などを参考にしていただきたい．

```
 1) Test plan identifier［テスト計画書 文書番号］
 2) Introduction［序文］
 3) Test items［テスト・アイテム］
 4) Features to be tested［テスト対象機能］
 5) Features not to tested［テスト非対象機能］
 6) Approach［アプローチ］
 7) Item pass/fail criteria［テスト・アイテム合否判定基準］
 8) Suspension criteria and resumption requirements［テスト中止基準とテスト再開要件］
 9) Test deliverables［テスト成果物］
10) Testing tasks［テストのタスク］
11) Environmental needs［テスト環境］
12) Responsibilities［責任範囲］
13) Staffing and training needs［人員計画およびトレーニング］
14) Schedule［スケジュール］
15) Risks and contingencies［リスクと対策］
16) Approvals［承認欄］
```

違うように感じています．テスト項目の進捗管理について，世間一般ではどのようなやりかたをしているんですか」



「不ぐあいの出かたで，順調かどうか違ってくると思った秋田君の感覚は正しい．いや，それを聞いて秋田君にテスト・チームを引き受けてもらってよかったと思ったよ．世間一般というか，日本のソフトウェアの品質管理の世界では『管理図』というものを使っているんだ．バグ管理図とかフォールト摘出状況管理図とも呼ばれているけどね（**図4**）．

要素項目としては，計画時の目標値として『テスト・ケース予定消化数』と『不具合検出推定数』があって，それからテスト実施フェーズになってからの実績値として『テスト・ケース実施数』と『不具合検出実績数』がある．テスト・ケース予定消化数とテスト・ケース実施数は残りのテスト項目数が時間軸に対していくつあるかの予定と実績を表現したものだから，基本的には右肩下がりのグラフになる．不ぐあい検出推定数と不ぐあい検出実績数はテストによって検出した不ぐあいの数，つまりバグ報告票（障害レポートや問題処理票など，呼びかたは企業や組織によりさまざま）の数を表現しているから，右肩上がりになる．

そういえば，バグの数がどうしてあらかじめわかっているんだと疑問に思うかもしれないね．過去のシステム開発の実績からだいたいのところを予測する場合もあるし，信頼度成長曲線というモデルを使って，ある程度不ぐあいが出たところで，残りの不ぐあいの数を推定するという方法もある．

じゃあ，管理図の基本的な見かたとして，四つのパターンを紹介しておこう」

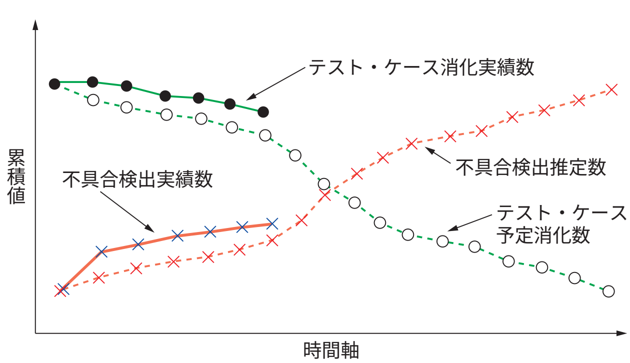


図4　管理図

開発中のプロダクトの品質を可視化する道具である．計画時の目標値である「テスト・ケース予定消化数」と「不具合検出推定数」，テスト実施時の実績値である「テスト・ケース実施数」と「不具合検出実績数」の四つの要素項目を持つ．中規模から大規模の開発で用いられており，メーカ系の品質保証部門でよく使われている．

ここでは**図4**の管理図の使いかたについて説明します．管理図は一般に，**図5**のようないくつかのパターンを示します．

1）順調

図5(a)ではテスト消化が予定どおりで，不ぐあいも予測どおりになっています．

2）リソース不足

図5(b)ではテスト消化が停滞していて，不ぐあいも見つけられていません．理由は二つ考えられます．一つ目は，テストを実施できるエンジニアの不足です．単に人数が足りないだけかもしれませんし，スキルが不十分な場合もあります．二つ目は，テスト環境に問題があるケースです．テスト用の試作機が足りなかったり，開発環境とテスト環境を共用していると，テストのために十分な設備が確保できなくなりがちです．

3）前工程に問題あり

図5(c)ではテスト消化が停滞していますが，不ぐあいは出続けています．これは，前工程での作り込みが不十分な状況です．不ぐあいが残ったまま，テストに移行したと考えられます．テストを続行しても意味がないどころか，やみくもにテストのリソースを浪費するだけかもしれません．こういった状況では，いったん前のプロセ

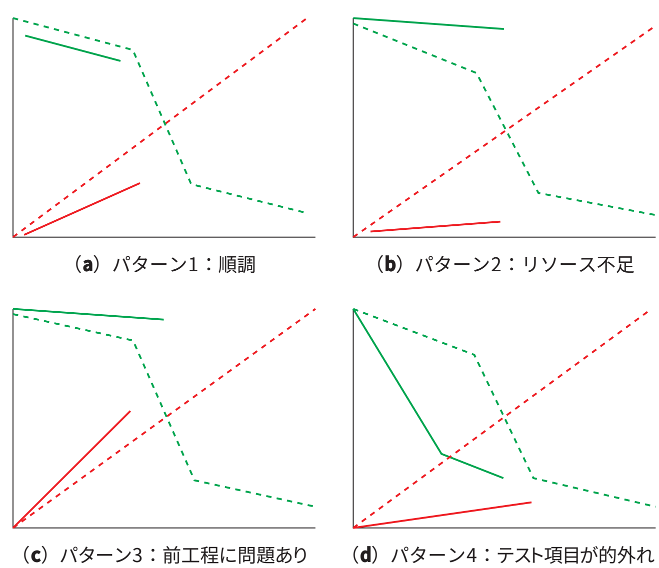


図5　管理図の使いかた

(a)はテスト消化が予定どおりで，不ぐあいも予測どおりの状態．順調にテストが進捗している．(b)はテスト消化が停滞していて，不ぐあいも見つけられていない状態．テストへのリソースが不足している．(c)はテスト消化が停滞しているが，不ぐあいは出続けている状態．前工程に問題がある．(d)はテスト消化は予定どおりか前倒しだが，不ぐあいが見つけられない状態．テスト項目が効果的に不ぐあいを摘出できない質であることがほとんど．

2

スである設計段階に差し戻す決断が必要となるでしょう．

4）テスト項目が的外れ

図5(d)ではテスト消化は予定どおりか前倒しですが，不ぐあいが見つけられません．一般的にはテスト設計を行ったエンジニアのスキルが低くて，的外れなテスト・ケースであるために不ぐあいが見つからない状況を表しています．また，単に数稼ぎにしかならないテスト項目数の数えかたをしていると，一つのテスト項目の確認ポイントそのものがあまり意味をなさないため，不ぐあいの潜む確率は当然低くなります．例えば1～100の入力に対する処理をテストするのに境界値分析を行わず，1～100のすべての入力を確認する項目を作成したような場合です．とはいえ，テスト項目の事前レビューが十分な優秀な組織であれば，設計品質が良好だからこそ，テストしても不ぐあいが出ないということはあります．

「実際にはシステム開発でほかの背景なんかも考慮しなくてはならないから，ステレオタイプ的に紹介した見かたが使えるとは限らない．管理図はあくまでもテスト進捗を管理するための道具だということを忘れないようにしてほしい．管理するための道具だから，問題パターンが発生したときに対策が打てないと意味がないからね．

例えば，テストで不ぐあいが思ったほど出ないのなら，テスト項目を再設計して項目そのものを充実させるとか，具体的な対策につなげるように使わないといけないんだよ」

Q7．チーム・メンバのメンタリングはどうする？

「今のところ3人のエンジニアにテスト・チーム専任メンバとしてがんばってもらってます．ただ，もともと実装をやりたがっていたエンジニアなので，どうやってこのままテストに対するモチベーションを維持したり，高めるためのメンタリングをすればいいのか，最近，悩んでるんです」

「テストに限った話ではないんだけど，メンタリングということばで大上段に構える必要はないんじゃないかな．まずは毎日きちんとコミュニケーションを取ることが大事だよ．とにかく顔を合わせて話す機会を設けないと．毎朝時間を決めて，10分程度でいいから全員の作業状況や昨日見つけた重要なバグの紹介，テスト環境や機材についての注意点（例えば，午前中は機器構成を変えてほしくないとか，データ設定をごそっと入れ替えたいけど，ほかのテスト実施に問題ないかどうかを事前確認するなど，ほかのメンバに影響すること）を話し合うことから始めてみてはどうだろう．ちなみに，工場の現場などではTBM（tool box meeting：道具箱の前で行うミーティングということから，この名が付いた）といって，毎朝チーム・メンバがコミュニケーションを取っているんだ．

テストのベテランといっしょにしごとをしてみるのもいいことだ．いわゆるOJT（on the job training）でテストのプロの技を伝授できればいいんだけど，われわれのようにテスト・チームが立ち上がったばかりだと，それも難しい．だとすると，テストにかかわる技術者のコミュニティに入ってみるとか，イベントに出かけて外部の知見を得る機会を増やすところから始めるといいよ」

国内のテストのコミュニティには「ソフトウェアテスト技術者交流会（TEF（テフ）：Testing Engineer's Forum）」があります[注4]．メーリング・リストを主たる活動の場として，1,000人を超えるテストに興味を持つエンジニアが参加しています．このTEFメンバで構成される実行委員会による，「ソフトウェアテストシンポジウム：JaSST」というイベントもあります[注5]．

Q8．バグ調査でメカ/エレキ技術者とうまくやるには？

「今の推敲フェーズのテストでは，シミュレータ上のプリンタやスキャナ，ファクシミリのハード・ドライバやコントローラ部のアーキテクチャの妥当性を確認しています．何か問題が見つかったとしても，ソフトウェアの開発部門が単独で調べればよい状況です．ですから不ぐあい調査は今のところ，わりとスムーズに進んでます．

これから作成フェーズに移行して，実機にソフトウェアを搭載する実機結合テストに突入すると，不ぐあい調査でメカやエレキのエンジニアとのコミュニケーションをうまく取る必要があると思ってます．そうしないと，過去の経験上，不ぐあいの1次調査でソフトウェアとハードウェアの原因の切り分けがうまくいかないと感じているからです．けっこう時間もかかりますし，コミュニケーションがうまくいかないと，ハードウェアとソフトウェアで不ぐあい調査の押し付け合いのようになることもありますから．

今回の開発のように『テスト・チーム』ができたとなると，

注4：テストに関する情報の総合ポータル・サイトで，TEFの主宰者である電気通信大学の西康晴氏により運営されている．http://www.swtest.jp/ を参照．

注5：次回は2006年1月30日，31日の2日間，東京都千代田区永田町の都市センターホテルにて「JaSST '06 in Tokyo」として開催される．http://www.jasst.jp/ を参照．

不ぐあい調査と名が付く作業なら何でもかんでもテスト・チームに押し付けられそうないやな予感もあります．

コミュニケーションのコツや気を付けたほうがよいことはありますか」



「メカやエレキのエンジニアとのコミュニケーションだけど，コミュニケーションの大前提として，相手のことばを理解するというのがある．“技術者”という人種は『こいつは，この技術に明るくないな』と思ったとたん，相手にしなくなる傾向がある．テストするエンジニアは，最低限，メカやエレキにかかわる不ぐあい報告書に書いてあるハードウェア側の原因や修正内容の概要が理解できるだけの知識を身につけておいてほしい．

複合機の世界だと，単体機能（コピー，プリンタ，スキャナ）は機能どおりに動いても，複合するとおかしな動作になることがよくある．複合動作する機能のとりまとめはコントローラ部，つまりソフトウェアで行っているんだけど，ハードウェアをどういったしくみでコントローラ部が制御しているのか，という『からくり』への理解がないと，不ぐあいのメカニズムが見えてこないんだ．

テスト・エンジニアは不ぐあいのにおいをかぎ分ける嗅覚（きゅうかく）が必要だし，それがあるからこそ存在感を示せるんだ．その嗅覚を養うためには，『組み込みシステム』がメカ・エレキ・ソフト全体をひっくるめて，どういったしくみで動いているのか，きちんと理解できるように知識と経験を身に付けないといけない．

そういった意味で，テスタに始まり，オシロスコープやディジタル・ハイコーダとか，必要に応じて計測器を使いこなす技術もテスト・エンジニアには求められると思うよ．計測器が扱えないと，そもそも不ぐあいの切り分けすら難しいことがあるからね．計測器の使いかた，そのときの注意点やコツなど，実際に学ぶべきことも多いよ．もし使いかたがわからなかったり，難しかったとしても，メカやエレキのエンジニアとうまくコミュニケーションが取れれば，いろいろと教えてくれると思う．

繰り返すようだけど，メカやエレキの設計経験がないからテスト・エンジニアに何をいってもむだだろう，などと思われないように，相手のことばを理解して，かつ相手を尊重する付き合いができるといいね．これは，開発部門の中でも同じで，開発者とテスト・チームの関係にも通じることなんだ」

Q9．テスト結果の記録はどうやるの？



「これまでテスト結果の記録といえば，Excelでまとめたテスト項目の一覧表にテスト結果を『OK/NG，担当者名，日付』といった項目として残すくらいしか行ってなかったんですけど，テスト・チームというからにはもっとシステマティックに記録しないといけないのでは，と考えています．実際，どこまで記録を残したほうがいいんでしょうか」



「何のためにテスト結果を記録するのか，だね．つまり，テストの目的によってテスト記録に残すべきことは変わってくる．テストの目的とひと口に言ってもいろいろとあって，例えば『不ぐあいを見つけたくてテストする』という場合もあれば，『テストによって動作に問題がないことを証明したくてテストする』という場合もある．また，『性能やメモリ，ディスク領域などのシステム・リソースの使用状況が要求仕様を満たしているかどうかを証明したくてテストする』ということもある．

まず，不ぐあいを見つけたくてテストすることを考えてみよう．不ぐあいを見つけたとき，その不ぐあいを再現できるようにテストを実行した記録が取れていないと意味がない．だから，テストを開始する前に，『不ぐあい調査』のための手段をしっかりと整えておく必要がある．

テストの手順はテスト項目に記載されているから，操作の流れは再現できる．けれど，実際どういったタイミングで，どういったデータの組み合わせにおいて操作をしたのか，テスト対象以外の機能の状態がどうだったのかまでは記録が残せていないことが多いんじゃないかな．

じゃあ，記録を残すためには何が必要かということになる．どういった操作をしたのか，タイム・スタンプを含めて記録するしくみを整えたり，前後のデータ設定や機能の状態を自動的にダンプするテスト用アプリケーションを用意したりする必要がある．計測器を使うことがあれば，ツール・ベンダが提供するソフトウェア・ツールを導入したり，自分たちでアプリケーションを作ることだってあるだろう．これだけのことをやろうと思うと，けっこう時間もコストもかかるし，習熟するためのトレーニング時間やツールのセットアップ，保守なんかもどうするか，検討が必要になる．

また，せっかく採取したテストの記録を，テスト結果としてどうやって保存するのかという問題もある．テスト用リポジトリとしての手段も準備しておかないと，結局，記

録は取りっぱなしになって，いざ不ぐあいが見つかって，ほかのテスト結果を含めた解析が必要になったときに，『あのデータが足りない』とか『記録が見つからない』なんていう騒ぎが起きてしまうんだ．だからこそ，テスト計画の段階から，テスト環境をしっかりと考えておかなくてはいけないんだよ．

次に，テストによって動作に問題がないことを証明したいのなら，どういった想定条件のもとでテストを実施したかをテスト項目や記録に残しているかがポイントになる．出荷してから不ぐあいが見つかって，『テストで動作確認を行ったことになっているのに，不ぐあいが見つかった』となると，ほかのテスト結果についても，その信頼性が疑われてしまうからね．だからこそ，そうならないためのしくみ作りがたいせつになるんだ．これはテスト項目と，テスト手順や判定内容を記載したテスト・ケースのようなドキュメントにしか結果を残せていない場合の問題と言い換えることができる．

とはいえ，テスト前後の状況をすべてドキュメントであらかじめ準備するというのも現実的には無理がある．不ぐあいを見つけるためのテストと同じように，できるだけ自動的にテスト結果を残すくふうもしてほしい．もちろん，状態遷移表やディシジョン・テーブルを使って，できる限りテストの条件を論理的に表現するくふうは，テスト・ケースを作成するときに行っていることが前提だ．

三つ目の要求仕様を満たしているかどうかを証明するためのテストだけど，これがけっこうくせ者だ．要求仕様を満たしているかどうか，計測器をつなげたり，動作をソフトウェアでトレースしてダンプしたりして測るわけだけど，測ることによって本来の性能が出なくなることがあるんだ．

例えば，ダンプを取るために特別なアプリケーションを作ったとする．そのアプリケーションが計測対象のシステムのCPUやメモリ・リソースを大きく占有してしまうなんてことは実はよくあることだ．ウォームアップからコピー・レディまでの時間を単純に測るだけなら，ストップ・ウォッチがあれば事足りる．でも，内部の状態遷移も含めて性能が出ているかどうかを測る場合には，どうしても内部処理を観察するためのソフトウェア・トレースなどの手段が必要になる．

こういった場合，あらかじめソフトウェア・トレースなしで，例えばストップ・ウォッチで実測値を調べておいてから，次にソフトウェア・トレースを起動して実測値と比較することで，トレースによる影響が出ていないかどうかをチェックできるようにしておくべきだろう．

計測器をつないでデータを観測しようとすると，回路構成が変わって，計測値が本来とは異なる結果になることもある．性能を測るときは，記録の妥当性を確保できるよう，あらかじめ十分に検討しておかないと，結果そのものが信用できなくなることを忘れないように」

Q10．バグの調査はどこまでやればいいの？


「テストでバグを見つけたときに悩ましいのは，どこまで調査を実施するかだと最近感じています．同じような不ぐあいが出ることがわかっている場合，不ぐあいのバリエーションの調査を一生懸命実施するメンバもいるのですが，そのせいでテストの進捗が滞ってしまうことが気がかりなんです」


「確かに，同じような不ぐあいの事象でもバリエーションがたくさんあると，どこまでテスト・チームで調査すればよいか悩むところだ．チーム・リーダの責務としては，見つかった不ぐあいがシステムに与える深刻度やテストの進捗への影響度から，どれくらい調査に時間を割いてよいかの基準をメンバに示すべきだろう．

考えかたとしては，今の推敲フェーズだとテスト進捗への影響より，その不ぐあいがアーキテクチャに対してどのくらい問題になりえるかという深刻度のほうに重きを置くべきだと思うよ．どれだけ不ぐあい調査をするかの基準に深刻度を使うのなら，テスト・メンバが不ぐあいを見つけたときに『重大，中度，軽度』と判断した3段階のそれぞれに対して，どれだけ不ぐあいの1次調査に時間を割くべきかの基準（上限値）を示すようにするんだ．例えば，重大は4時間，中度は2時間，軽度は1時間といったようにね．忘れちゃいけないのは，この1次調査を終えてから，さらにバグ管理票を起票するためのデータの整理だとか，起票作業にも時間がかかることだ．

次に，不ぐあいにバリエーションがあるときの調査の線引きについて少し説明しよう．バグ管理票の基本は，『1件のバグ管理票には一つの不ぐあいを書く』ということなんだけど，それを可能とするためには不ぐあいが発生するメカニズムをある程度つかんでおかなくてはならないんだ（**図6**）．

不ぐあいとひと口に言っても因果関係があって，この図で説明すると，故障とは『ソフトウェアが期待どおりに動作しない現象』を指す．この故障の原因となるのが欠陥で，

プログラム内の欠陥や誤りが元となる．欠陥の元になるのがエラーで，実装時の誤植や仕様の読み違え，入力ミスなどがその原因になるといった関係がある．

例えば，紙切れやジャムなどの異常が検出されてなくて，『コピー可能』状態にもかかわらず，コピー用紙をセットしてコピー・ボタンを押してもコピーできなければ，故障ということになる．原因としては，本来紙切れなどの異常な状態であるのに，異常状態を検出できていないという欠陥があるのかもしれない．この異常状態を検出できないのは，プログラムで異常状態を判定するアルゴリズム部に記述ミス，つまりエラーがあるからかもしれない．こういったように，かならず因果関係があるはずなんだ．

テストで不ぐあいを見つけたとき，その不ぐあいという事象にどんな因果関係があるのかを意識したうえで調査しないと，やみくもな調査となってしまうし，結果として1件のバグ管理票に複数の故障や欠陥が記述されてしまうことにもなりかねないんだ」

＊　　＊　　＊

秋田君と大山さんのやりとりの紹介はこれまでとなります．ここまで紹介したテスト計画で五つ，テスト管理で五つ，合計10の問題とその解決案は，現実に起きうることのほんのわずかな例に過ぎません．

筆者としては，本記事を通して，ソフトウェア開発におけるテストとは，単にソフトウェアやシステムの動作確認にとどまるものではなく，体系的に品質を評価して保証するために実施するプロセスであることを少しでも理解していただければ幸いです．

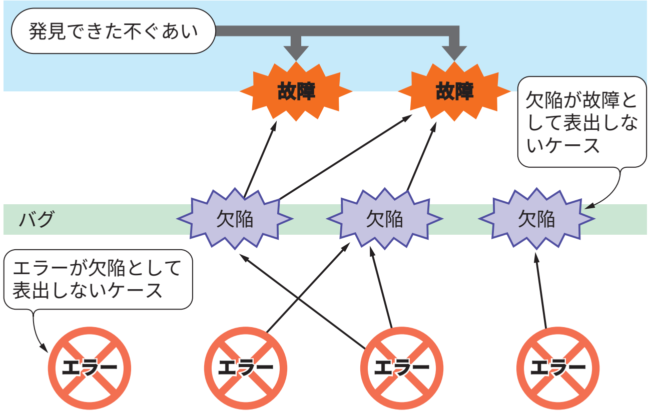


図6　不ぐあいが発生するメカニズム(9)

不ぐあいの発生メカニズムを故障，欠陥，エラーの三つの従属関係で表わしている．故障を「ソフトウェアが期待どおりに動作しない現象」とした場合，故障の原因が欠陥であり，その元がエラーとなる．故障は単独の欠陥やエラーのみで引き起こされるのではなく，それぞれの組み合わせにより故障につながることのほうが多い．

参考・引用*文献

(1) ロジャーS.プレスマン；実践ソフトウェアエンジニアリング，日科技連出版，2005年2月．
(2) ヤコブソン，ブーチ，ランボー；UMLによる統一ソフトウェア開発プロセス，翔泳社，2000年3月．
(3) バリー・ベーム，リチャード・ターナー；アジャイルと規律，日経BP社，2004年8月．
(4) 2002年度 事務機器関連調査報告書，社団法人 ビジネス機械・情報システム産業協会，2002年．
(5) 2004年度 事務機器関連調査報告書，社団法人 ビジネス機械・情報システム産業協会，2004年．
(6) DTP&印刷しくみ事典——印刷メディアディレクター&デザイナーのためのグラフィックバイブル，ワークスコーポレーション，2005年1月．
(7) 保田勝通；ソフトウェア品質保証の考え方と実際，日科技連出版社，1995年11月．
(8) 居駒幹夫(SESSAME編)；第3回組み込みソフトウェア技術者管理者向けセミナーテキスト 組み込みソフトの品質管理編，2004年．
(9)* 大西建児；ステップアップのためのソフトウエアテスト実践ガイド，日経BP社，2004年7月．
(10) 大西建児；特集1 ソフトウェアテスト入門，ソフトウェアテストPRESS，vol.1，技術評論社，2005年6月．
(11) S-openオフショア開発研究会；ソフトウエア開発オフショアリング完全ガイド，日経BP社，2004年10月．
(12) Cem Kaner，James Bach，Bret Pettichord；ソフトウェアテスト293の鉄則，日経BP社，2003年4月．

おおにし・けんじ
(株)豆蔵 シニアコンサルタント

<筆者プロフィール>
大西建児．国内電機メーカ，外資系通信機器ベンダでのソフトウェア・テストの経験を生かし，ソフトウェア・テストを中心にソフトウェア品質改善全般のコンサルティング活動に従事．著書に「ステップアップのためのソフトウエアテスト実践ガイド」(日経BP社)，訳書に「基本から学ぶテスト・プロセス管理」(日経BP社)などがある．